License Plate Recognition Project

cny123222 zaan-lee ephuon gg-bond12 ykxin

January 15, 2025

1 Experiment Overview (实验概览)

License Plate Recognition (**LPR**) is a critical technology widely applied in intelligent transportation systems, including toll collection, parking management, and traffic monitoring. However, LPR poses several challenges due to factors such as varying lighting conditions, and complex backgrounds. To address these difficulties, we **explored various approaches** to identify the most effective solution for robust and accurate recognition.

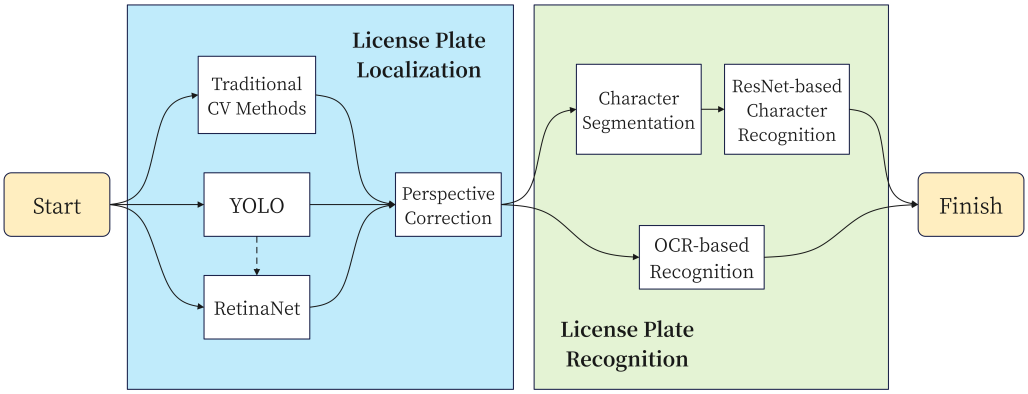


Figure 1 Overall Workflow of the LPR Project

In our experiment, we examined a range of methods for license plate localization and recognition.

- For **localization**, we utilized **traditional computer vision techniques** alongside deep learning-based object detection models such as **YOLO** (You Only Look Once) and **RetinaNet**. After extracting the license plate region, **perspective correction** was applied to reduce distortion and improve downstream recognition accuracy.
- For **recognition**, we experimented with both **segmentation-based** and **end-to-end recognition techniques**. Specifically, we tested a **character segmentation approach** followed by **ResNet-based character classification** and an end-to-end **OCR-based recognition pipeline**. Additionally, we evaluated **CRNN** (Convolutional Recurrent Neural Network) for its potential in sequence-based recognition tasks.

After thorough comparison and evaluation, we selected a combination of **YOLO, RetinaNet, and OCR-based recognition** as our final solution. This approach demonstrated excellent performance in rec-

ognizing **blue, green, and yellow** license plates commonly seen **in China**, achieving high accuracy on the **CCPD dataset**.

As the culmination of our project, we developed a graphical user interface (**GUI**) application that integrates our selected pipeline into a user-friendly program capable of performing real-time license plate recognition. This solution not only demonstrates the effectiveness of our method but also provides practical utility for real-world applications.

2 Solution Approach (解决思路)

2.1 LP Localization

Accurately localizing the license plate (LP) within an image is the foundational step for any LP recognition system. This task is particularly challenging due to variations in plate sizes, orientations, and background clutter in real-world scenarios. To address these challenges, we explored multiple approaches, including traditional CV techniques and modern deep learning-based methods, to compare their performance and robustness.

2.1.1 Traditional CV Approaches

Traditional computer vision (CV) methods remain widely utilized for LP localization due to their simplicity and lack of dependence on extensive labeled datasets. In this subsection, we provide a detailed, step-by-step description of our CV-based approach.

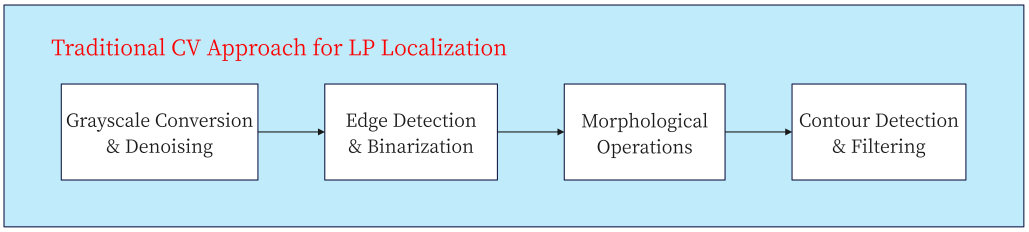


Figure 2 Workflow of Traditional CV Approaches

Step1 (Grayscale Conversion and Denoising) : The initial step converts the input image into grayscale, simplifying computations and emphasizing essential features while reducing noise. Gaussian blurring is employed to smooth the image by suppressing high-frequency noise. This produces a cleaner grayscale image that highlights key structures necessary for subsequent processing.

```
image = cv2.cvtColor(image, cv2.COLOR_RGB2GRAY) # Grayscale conversion
image = cv2.GaussianBlur(image, (3, 3), 0) # Gaussian blurring
```

Step2 (Edge Detection and Binarization) : To detect the LP, edges are extracted using the Sobel operator in the x-direction, which focuses on vertical edges crucial for LP identification. The resulting edge-detected image is then binarized using Otsu's thresholding method. This dynamic thresholding separates the foreground (LP features) from the background, producing a clear binary image suitable for further analysis.

```
Sobel_x = cv2.Sobel(image, cv2.CV_16S, 1, 0) # Edge detection using Sobel
```

```
img = cv2.convertScaleAbs(Sobel_x) # Convert into 8-bit image
_, image = cv2.threshold(image, 0, 255, cv2.THRESH_OTSU) # Binarization using Otsu's
    thresholding
```

Step3 (Morphological Operations) : Morphological transformations are applied to refine the binary image by enhancing relevant details and suppressing noise. A morphological closing operation fills gaps within edges, improving edge connectivity. This is followed by iterative dilations and erosions using carefully designed structuring elements to consolidate LP features and filter out irrelevant noise. A final median blur smoothens the processed image, preparing it for contour detection.

```
# Horizontal structuring element
kernelX = cv2.getStructuringElement(cv2.MORPH_RECT, (17, 5))
# Morphological closing
image = cv2.morphologyEx(image, cv2.MORPH_CLOSE, kernelX, iterations=3)
# Morphological Operations
kernelX = cv2.getStructuringElement(cv2.MORPH_RECT, (50, 1))
kernelY = cv2.getStructuringElement(cv2.MORPH_RECT, (1, 20))
image = cv2.dilate(image, kernelX) # Dilation
image = cv2.erode(image, kernelX) # Erosion
image = cv2.erode(image, kernelY) # Erosion
image = cv2.dilate(image, kernelY) # Dilation
image = cv2.medianBlur(image, 21) # Median blur
```

Step4 (Contour Detection and Filtering) : Contours are extracted from the processed image to locate potential LP regions. Each contour is analyzed and filtered based on its aspect ratio. Contours matching the characteristics of an LP are cropped.

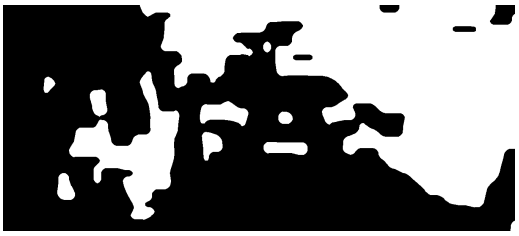
```
# Contour detection
contours, hierarchy = cv2.findContours(image, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
for contour in contours: # Check each contour
    rect = cv2.boundingRect(item) # Get bounding rectangle
    x, y, width, height = rect
    # Filter by aspect ratio
    if (width > (height * 2.5)) and (width < (height * 5)):
        plate_image = origin_image[y:y + height, x:x + width] # Crop potential LP
```

This pipeline effectively isolates the license plate (LP) by leveraging geometric and morphological features. Figure 3 illustrates the intermediate results at each stage of the process, culminating in the successful extraction of the LP. These visualizations provide insight into the transformations applied during each step, enhancing our understanding of the pipeline's progression.

Initial Image

After Step1: Grayscale Conversion

After Step2: Edge Detection and Binarizatio



After Step3: Morphological Operation

After Step4-1: Contour Detection

After Step4-2: Contour Filtering

Figure 3 Intermediate Results of the Traditional CV Pipeline

2.1.2 YOLO

The You Only Look Once (YOLO) series is a family of deep learning object detection algorithms known for real-time performance and high accuracy. YOLO models predict bounding box coordinates and class probabilities in a single pass, combining feature extraction, bounding box regression, and class prediction in a unified network.

In this work, we employed a pre-trained YOLOv5 model specifically fine-tuned for license plate (LP) detection. The workflow involves feeding the input image into the YOLO model to obtain bounding box predictions.

```
model = yolov5.load('keremberke/yolov5n-license-plate') # Load YOLOv5 model
results = model(image, size=640) # Perform inference
predictions = results.pred[0] # Get detection results
box = predictions[0, :4].cpu().numpy().astype(int) # Extract bounding box coordinates
x1, y1, x2, y2 = expand_bbox(image, box, expand_ratio) # Expand bounding box
plate_image = image[y1:y2, x1:x2] # Crop license plate image
```

Figure 4 shows some examples of YOLO-based LP localization, where the bounding box accurately highlights the LP. YOLOv5's fast inference speed and high detection accuracy make it particularly suitable for real-time scenarios.

Figure 4 Examples of YOLO-based LP Localization

However, since YOLO predicts rectangular bounding boxes, it cannot directly locate the four precise corner points of the license plate. Without applying perspective correction, this limitation may negatively impact recognition accuracy. To address this, we expand the predicted bounding box by 20%, allowing for further processing and refinement.

2.1.3 RetinaNet

RetinaNet is a widely-used one-stage object detection algorithm known for its balanced performance between speed and accuracy. Unlike traditional rectangular bounding box detectors, RetinaNet can be adapted for more flexible tasks, such as corner point detection. By leveraging its focal loss mechanism and feature pyramid network (FPN), RetinaNet effectively detects objects at varying scales, making it suitable for fine-grained localization tasks like LP corner point detection.

In this work, we adapted RetinaNet to predict the four corner points of license plates, enabling precise localization for perspective correction. Figure 5 demonstrates the results of RetinaNet applied to corner point localization.

Figure 5 Examples of LP Localization Using RetinaNet

However, directly applying RetinaNet to the entire image for LP corner detection often introduces inaccuracies due to background clutter and varying plate sizes. To address these challenges, we **innovatively combined YOLO and RetinaNet**. YOLO is first used to identify the general region of the license plate. Then, RetinaNet is applied within the YOLO-predicted bounding box to extract the precise corner points. This hierarchical approach mitigates errors from applying RetinaNet directly to the whole image, leading to improved robustness and accuracy in localization. The combined method enhances the quality of perspective correction, forming a strong foundation for downstream recognition tasks.

2.2 Color Recognition

Color recognition is an essential step in license plate recognition (LPR) systems, particularly in scenarios where license plates are categorized by color to indicate different vehicle types or usage regulations. In this work, we implemented a straightforward yet effective method to classify license plates into three common colors in China: **yellow, blue, and green**.

Our method leverages the **HSV (Hue, Saturation, Value) color space**, which separates color information (hue) from intensity (value). This separation makes the HSV space more robust to lighting variations compared to the RGB color space. Specifically, we defined fixed HSV ranges for each target color:

- **Yellow**: $H \in [20, 30], S \in [100, 255], V \in [100, 255]$
- **Blue**: $H \in [100, 130], S \in [100, 255], V \in [100, 255]$
- **Green**: $H \in [40, 80], S \in [100, 255], V \in [100, 255]$

The process involves the following steps:

- Convert the input license plate image to HSV color space.
- Apply color masks to isolate pixels within the predefined HSV ranges for each color.
- Count the number of pixels for each color mask.
- Identify the dominant color by comparing the pixel counts.

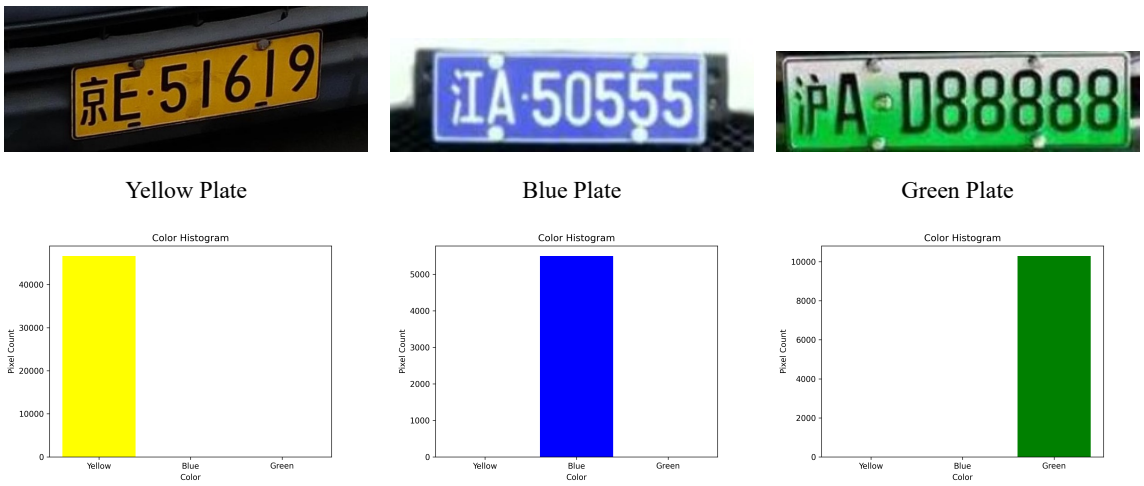
```python
def detect_plate_color(image):
    # Convert to HSV color space
    hsv_image = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)

    # Define HSV ranges for yellow, blue, and green
    yellow_lower, yellow_upper = np.array([20, 100, 100]), np.array([30, 255, 255])
    blue_lower, blue_upper = np.array([100, 100, 100]), np.array([130, 255, 255])
    green_lower, green_upper = np.array([40, 100, 100]), np.array([80, 255, 255])

    # Create masks for each color
    yellow_mask = cv2.inRange(hsv_image, yellow_lower, yellow_upper)
    blue_mask = cv2.inRange(hsv_image, blue_lower, blue_upper)
    green_mask = cv2.inRange(hsv_image, green_lower, green_upper)

    # Count pixels for each color
    color_counts = {
        'Yellow': cv2.countNonZero(yellow_mask),
        'Blue': cv2.countNonZero(blue_mask),
        'Green': cv2.countNonZero(green_mask),
    }

    # Determine the dominant color
    dominant_color = max(color_counts, key=color_counts.get)
    return dominant_color
```

Figure 6 presents examples of license plates in yellow, blue, and green. The corresponding histograms demonstrate the pixel distribution for each color mask, validating the accuracy of our method in identifying the dominant color.

From the figure, it is evident that this method effectively distinguishes between the three target colors under various lighting conditions. By leveraging the robustness of the HSV color space, we achieved high accuracy in identifying the dominant color of license plates, which is critical for categorization and further processing in LPR systems.

2.3 Perspective Correction

Perspective distortion is a common issue in license plate recognition, as plates are often captured at oblique angles or in skewed orientations. Accurate perspective correction ensures that the extracted license



Yellow Plate | Blue Plate | Green Plate

Yellow Histogram | Blue Histogram | Green Histogram

Figure 6 Examples of Color Recognition Results and Corresponding Histograms

plate region is geometrically rectified, facilitating accurate recognition in subsequent steps. In this work, we utilized the four corner points of the license plate predicted by the RetinaNet model to perform **perspective transformation**. This approach precisely **maps the plate region to a standard rectangular shape**.

The process involves defining a mapping between the detected corner points in the original image and a predefined rectangular template. Using OpenCV's `cv2.getPerspectiveTransform` function, we computed the transformation matrix, which was subsequently applied to the extracted plate region via `cv2.warpPerspective`. The final output is a corrected license plate image suitable for recognition.

```
# Extract bounding box and crop the plate region
x1, y1, x2, y2 = b[0], b[1], b[2], b[3]
img_box = img[y1:y2 + 1, x1:x2 + 1, :]

# Compute relative coordinates of the corner points
new_x1, new_y1 = b[9] - x1, b[10] - y1
new_x2, new_y2 = b[11] - x1, b[12] - y1
new_x3, new_y3 = b[7] - x1, b[8] - y1
new_x4, new_y4 = b[5] - x1, b[6] - y1

# Define the source and target points for transformation
points1 = np.float32([[new_x1, new_y1], [new_x2, new_y2], [new_x3, new_y3], [new_x4,
    new_y4]])
points2 = np.float32([[0, 0], [188, 0], [0, 48], [188, 48]])

# Compute the transformation matrix and apply perspective correction
matrix = cv2.getPerspectiveTransform(points1, points2)
result = cv2.warpPerspective(img_box, matrix, (188, 48))
```

This technique effectively normalizes the perspective of the license plate region, enabling robust recognition even under challenging capture conditions. Figure 7 illustrates examples of perspective correction

applied to skewed license plate images, demonstrating the efficacy of the approach.

Before Perspective Correction

After Perspective Correction

Before Perspective Correction

After Perspective Correction

Figure 7 Examples of Perspective Correction Results

2.4 LP recognition

After localizing and correcting the license plate, the next step involves recognizing the characters on the plate. This process plays a pivotal role in the overall LPR system, converting the visual representation of the license plate into interpretable text. We explored two primary approaches for LP recognition: **segmentation-based character recognition** and **end-to-end recognition**.

2.4.1 Character Segmentation

The process of character segmentation plays a crucial role in license plate recognition (LPR) systems, as it involves isolating individual characters from the detected license plate region for subsequent recognition. Our approach combines image processing techniques and projection-based analysis to achieve precise character segmentation. Below, we outline the key steps involved.

Step1 (Preprocessing and Binarization) : The input license plate image is first converted to grayscale to reduce computational complexity. A binarization step is applied using Otsu's method to separate the foreground (characters) from the background. If the detected character color is white, the binary image is inverted to ensure consistency.

Step2 (Morphological Operations) : To bridge gaps in broken characters, dilation operations are applied. An additional dilation step is performed to address disjointed areas, particularly for Chinese characters, which often exhibit complex structures.

Step3 (Horizontal Projection for Plate Region Localization) : A horizontal projection histogram is computed by summing the pixel intensities along rows. This histogram helps identify the vertical boundaries of the license plate within the image, isolating the relevant region for further processing.

Step4 (Vertical Projection for Character Segmentation) : Within the localized license plate region, a vertical projection histogram is generated by summing pixel intensities along columns. Peaks and valleys in this histogram indicate the boundaries between adjacent characters. These boundaries are used to segment the license plate into individual character regions.

Step 5 (Character Refinement and Segmentation) : The vertical segmentation results are refined by identifying the maximum center distance between adjacent character regions. This ensures consistent spacing for character extraction. The segmented characters are then extracted into separate image regions for subsequent recognition.

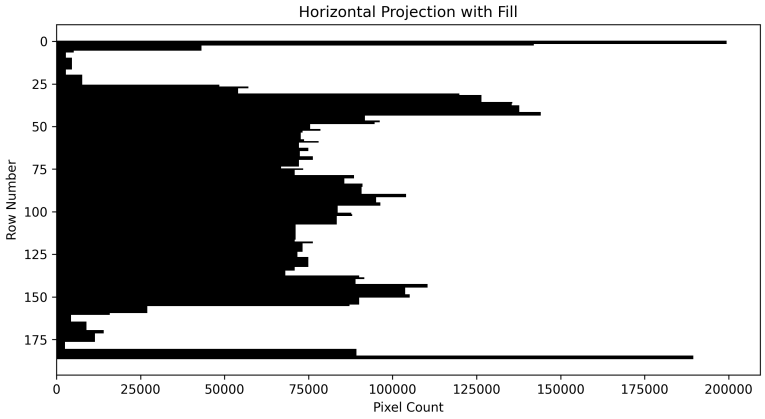
The core implementation steps are summarized in the following Python code:

```
# Load and preprocess the license plate image
gray_image = load_image(image)
bi_image = binarize_image(gray_image)

# Detect character color and invert if necessary
if detect_character_color(bi_image) == 'white':
    bi_image = cv2.bitwise_not(bi_image)

# Morphological operations to connect characters
binary_image = connect_characters(bi_image)
binary_image = connect_characters_double(binary_image)

# Horizontal projection for plate localization
horizontal_proj, bi_image = horizontal_projection(binary_image, bi_image)

# Vertical projection for character segmentation
vertical_proj = vertical_projection(horizontal_proj)

# Identify split positions and segment characters
character_regions = find_split_positions_vertical(vertical_proj)
max_distance = calculate_max_center_distance(character_regions)
segments = vertical_segmentation(bi_image, horizontal_proj, max_distance)

# Display segmented characters
show_segmented_characters(segments)
```

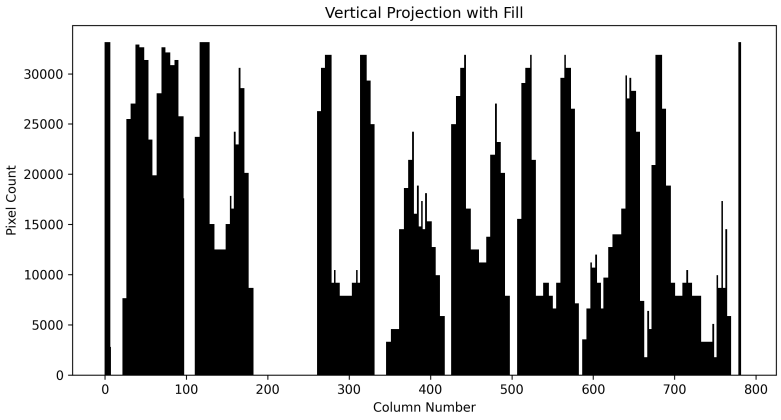
Figure 8 provides an example of our character segmentation pipeline. The input license plate image (a) undergoes horizontal projection (c) to determine the plate boundaries, followed by vertical projection (d) to isolate individual characters. Finally, segmented character results are shown in (b).

(a) Input License Plate Image

(b) Segmented Characters



(c) Horizontal Projection



(d) Vertical Projection

Figure 8 Example of Character Segmentation Process

This segmentation pipeline effectively isolates each character from the license plate, ensuring high-quality inputs for subsequent character recognition.

2.4.2 ResNet-based Character Recognition

In our license plate recognition (LPR) system, we adopted **ResNet-18** as the backbone for character recognition due to its balanced performance in accuracy and computational efficiency. We trained three distinct networks tailored to the unique requirements of license plate character classification: one for recognizing Chinese characters, another for alphanumeric characters, and a third network specifically for English letters.

A notable aspect of our approach was the separate training of a network exclusively for English letter recognition, leveraging the fact that the second character of Chinese license plates is always an English

letter. This specialization aimed to enhance accuracy by reducing class overlap in the classification task.

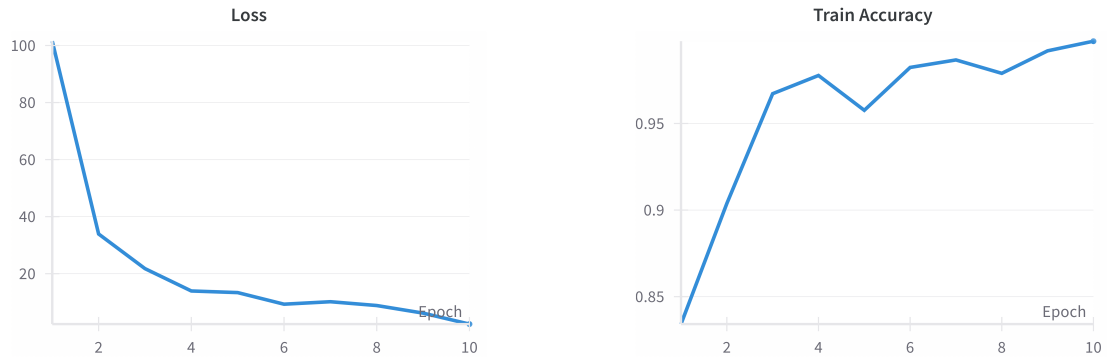
The training procedure for the Chinese character network used a non-pretrained ResNet-18 model trained for 10 epochs, while the alphanumeric and English networks were initialized with pretrained weights and fine-tuned for 5 epochs. The final training accuracies on the training set were as follows: **0.9879** for Chinese characters, **0.9942** for alphanumeric characters, and **0.9944** for English letters.

Table 1 Training Results of ResNet-based Character Recognition Networks

Network Type	Pretraining	Epochs	Accuracy
Chinese Character	No	10	0.9879
Alphanumeric Character	Yes	5 (fine-tune)	0.9942
English Letter	Yes	5 (fine-tune)	0.9944

We also performed **k-fold cross-validation** to assess the generalization performance. Taking the Chinese character network as an example, the cross-validation accuracy was **0.9545**, indicating minimal overfitting and demonstrating the robustness of the training process.

Figures 9a and 9b show the training loss and accuracy curves for the Chinese character recognition network. The steady decrease in loss and the convergence of accuracy near the 10th epoch reflect effective model training.



(a) Training Loss Curve (b) Training Accuracy Curve

Figure 9 Performance Curves for the Chinese Character Recognition Network

This multi-network training strategy, coupled with the specialized English letter network, demonstrates the effectiveness of ResNet-18 in addressing the diverse character recognition requirements in license plate systems while maintaining high accuracy and robustness.

2.4.3 OCR-based Recognition

Optical Character Recognition (OCR) offers a significant advantage over traditional character segmentation methods by **avoiding the complexities of manually segmenting characters**. This enhances the robustness of recognition, particularly in challenging scenarios such as distorted or low-quality images.

In our experiments, we tested several OCR solutions, including **PaddleOCR**, **Tesseract OCR**, and **RapidOCR**, among others. After extensive evaluation, we selected **RapidOCR** as the final OCR engine

for our license plate recognition system. RapidOCR demonstrated exceptional performance in recognizing license plates under complex conditions, including skewed angles, varying lighting, and noisy backgrounds.

RapidOCR's superior recognition capabilities stem from its powerful text detection and recognition pipeline, which includes advanced text segmentation algorithms and language modeling for context-aware recognition. These features make it particularly well-suited for recognizing alphanumeric sequences on license plates, even when the plates are partially obscured.

Table 2 illustrates several examples of license plate recognition using RapidOCR. The results highlight the model's accuracy and robustness across various license plate images.

Table 2 Examples of License Plate Recognition using RapidOCR

License Plate Image	OCR Recognized Result
	皖 AS0747
	皖 MK988X
	皖 AZIY26
	皖 AH924G

The integration of RapidOCR into our LPR system significantly enhanced recognition accuracy and robustness, especially in scenarios where traditional segmentation-based methods struggled. By leveraging the strengths of advanced OCR techniques, we ensured reliable performance across a wide range of license plate types and environmental conditions.

2.5 Graphical User Interface (GUI)

To provide a user-friendly platform for real-time license plate recognition, we developed a Graphical User Interface (GUI) using Python's **Tkinter** library. The interface integrates all key components of our pipeline, including license plate localization, perspective correction, character segmentation, color recognition, and OCR-based recognition.

The GUI is designed to be intuitive and functional. Users can perform the following actions:

- **Select an Image:** The "选择图片" button allows users to upload a car image for processing.
- **Run Recognition:** The "识别图片" button triggers the recognition pipeline. It includes license plate localization using YOLO and RetinaNet, and recognition using OCR-based methods.
- **View Results:** The interface displays the original image, the localized license plate, and the final

Figure 10　License Plate Recognition GUI

recognition result. The recognized license plate text is displayed in its corresponding background color (blue, green, or yellow), providing visual clarity.

Figure 11　GUI Examples: End-to-End License Plate Recognition Results

This GUI demonstrates a complete end-to-end implementation of our license plate recognition system, offering both technical robustness and ease of use. Figure 10 shows the interface, highlighting its modular structure and clear result visualization. Additionally, Figure 11 showcases two examples where the GUI accurately processes license plates, from localization to recognition.

3　Experimental Results (实验结果)

In our study, we ultimately selected a combination of **YOLO, RetinaNet, and RapidOCR** for the license plate recognition pipeline. The rationale behind this choice will be elaborated in the subsequent section. To evaluate the effectiveness of this approach, we conducted experiments on the **CCPD dataset**, which contains diverse and complex license plate images.

From the validation set, we randomly sampled **997 images** for testing. Figure 12 presents a collage of several sample images from the CCPD dataset, showcasing its variability in terms of lighting conditions,

plate types, and backgrounds. Our final pipeline achieved a recognition accuracy of **96.09%**, demonstrating its robustness and effectiveness in real-world scenarios.

Figure 12 Sample Images from the CCPD Dataset

The high accuracy achieved validates the strength of the combined YOLO+Retina+RapidOCR approach, effectively balancing precision and robustness in license plate recognition tasks.

4 Analysis and Discussion (分析与思考)

4.1 Why YOLO + Retina + RapidOCR?

4.1.1 Limitations of Traditional Methods

During our experiments, we found that traditional computer vision approaches faced significant challenges:

- **Sensitivity to Environment:** Traditional methods are highly sensitive to environmental factors such as lighting conditions, shadows, and reflections. These factors can severely degrade performance, particularly in images with non-uniform lighting or glossy surfaces.
- **Dependency on Parameter Tuning:** The effectiveness of traditional methods often depends on manual parameter tuning (e.g., kernel sizes for morphological operations, threshold values for binarization). This makes the pipeline brittle and difficult to generalize across diverse scenarios.
- **Impact of Plate and Vehicle Color:** Variations in license plate colors and vehicle paintwork can introduce additional noise, complicating segmentation and recognition tasks.
- **Uncertainty in Results:** Traditional methods are prone to errors in character segmentation, especially in cases with low contrast or overlapping characters. These uncertainties can propagate through the pipeline, leading to unreliable recognition.

4.1.2 Advantages of Our Method

The decision to use the **YOLO + Retina + RapidOCR** combination stems from a balance between accuracy, robustness, and computational efficiency:

- **YOLO** effectively localizes license plates with high accuracy and speed, providing a bounding box that reduces irrelevant regions and enhances downstream processing.
- **RetinaNet** accurately identifies the four corner points of the license plate, enabling precise perspective correction and rectification for subsequent recognition.
- **RapidOCR** eliminates the need for complex character segmentation, offering a robust end-to-end pipeline that adapts well to challenging scenarios, such as varying lighting conditions and cluttered backgrounds.

By adopting a modern pipeline integrating **YOLO**, **RetinaNet**, and **RapidOCR**, we overcame many of these limitations. The use of deep learning-based models allowed us to achieve higher accuracy and robustness while reducing dependency on manual parameter tuning. This combination enabled consistent performance across varied conditions, as demonstrated by the high recognition accuracy on the CCPD dataset.

4.2 Challenges in Character Segmentation

During our exploration of traditional CV methods, we came across our biggest challenge in the process of character segmentation:

- **Boundary Noise from Retinal Corner Points:** The corner points detected by RetinaNet often included portions of the license plate's outer frame. This caused the input plate region to contain unwanted boundary elements, leading to high peaks in the horizontal and vertical projection histograms, which in turn hindered character segmentation accuracy.
- **Split Characters in Chinese Plates:** For Chinese license plates, horizontally structured Chinese characters (e.g., "贛") were sometimes misinterpreted as two separate characters during segmentation. This issue arose from the unique geometry of these characters.

To address these challenges, we introduced several creative solutions:

- For boundary noise, we adopted dual horizontal and vertical projection histograms and width-based detection to refine the extracted plate region.
- To handle split Chinese characters, we adopted morphological operations, coupled with width-based detection, to improve segmentation precision and ensure the integrity of characters.

4.3 Observations from CRNN Experiments

We also experimented with **CRNN (Convolutional Recurrent Neural Network)**, a popular end-to-end OCR model. However, the results were suboptimal. Several factors likely contributed to its underperformance:

- **Lack of Perspective Correction:** Plates fed into the CRNN model lacked proper rectification, leading to distorted input images that negatively impacted recognition accuracy.
- **High Variation in Plate Content:** The CCPD dataset includes diverse colors, and styles, which may have hindered the model's ability to generalize effectively.
- **Inadequate Data Augmentation:** The CRNN model may have benefited from additional augmentation techniques to simulate various real-world conditions and improve robustness.

4.4 Further Considerations

Despite achieving high recognition accuracy, there are still areas where our system can be improved:

- **Limitations in Plate Types:** The current system is designed to recognize standard civilian license plates, primarily blue, green, and yellow plates. It is unable to handle special license plates, such as those used by police vehicles, military vehicles, or diplomatic cars, which often feature distinct designs or patterns.
- **Restricted Color Recognition:** While our system effectively identifies common plate colors, extending support to other colors would enhance its generalizability.
- **Challenges with Difficult Plates:** Plates that are severely blurred, obscured, or tilted at extreme angles remain difficult to process accurately. Although our system integrates perspective correction and robust OCR, highly challenging conditions still impact performance.
- **Speed Optimization:** The current recognition pipeline, while accurate, has room for improvement in terms of processing speed. This is especially critical for real-time applications such as traffic monitoring or toll collection, where high throughput is required.

Future work can address these limitations by incorporating specialized models for unique plate types, extending color detection capabilities, and enhancing the robustness of recognition under extreme conditions. Additionally, further optimization of the pipeline's computational efficiency, including hardware acceleration techniques, could significantly improve recognition speed without compromising accuracy.

5 Reflection and Conclusion (实验感想)

5.1 Learning from Experiments

Through this project, we gained valuable insights into **the strengths and limitations of combining traditional computer vision techniques with neural networks**. Traditional methods excelled in tasks like color recognition with rule-based algorithms, while deep learning models like ResNet and RapidOCR proved indispensable for accurate and adaptable character recognition.

The project **deepened our understanding of computer vision principles**, including morphological operations, edge detection, and neural networks. We also **improved our skills in research and application**, effectively integrating academic knowledge into practical scenarios.

Collaboration played a key role, reinforcing the importance of **teamwork and problem-solving** in scientific research. Additionally, the challenges of debugging, optimization, and validation taught us the value of perseverance and critical thinking, fostering our growth as researchers.

5.2 Conclusion

In conclusion, this project successfully **demonstrated the power of combining traditional computer vision techniques with neural networks** to create an efficient and accurate license plate recognition system. By utilizing traditional methods for tasks like color recognition and employing neural networks for character segmentation and recognition, we achieved a harmonious balance between robustness and precision. This hybrid approach enabled us to capitalize on the unique strengths of both methodologies, achieving a high level of recognition accuracy.

Beyond the technical achievements, this project served as a valuable learning experience that enriched our knowledge of computer vision, reinforced our collaborative skills, and instilled in us a deeper appreciation for the research process. These lessons will undoubtedly guide us in future academic and professional endeavors.

6 References

https://huggingface.co/keremberke/yolov5n-license-plate
https://github.com/zxm97/RetinaFace_LPRNet_TensorRT
https://rapidai.github.io/RapidOCRDocs/